

STATISTICAL GRAPHICS IN R

Terry A. Cox, M.D., Ph.D.
National Eye Institute

Course Outline

1. Websites and documentation
2. Principles for creating effective statistical graphics
3. Creating, customizing, and exporting R graphics
4. R graphics in computer presentations

URLs

- <http://www.r-project.org/>

The source for R software and documentation.

- <http://www.cs.wisc.edu/~ghost/>

Ghostscript and GSview, software for interpreting and viewing PostScript files.

- <http://www.ggobi.org/>

GGobi is a data visualization system for viewing high-dimensional data.

- <http://desktoppub.about.com/cs/graphicformats/index.htm>

Information on graphics formats.

- <http://www.personal.psu.edu/faculty/c/a/cab38/ColorBrewerBeta.html>

ColorBrewer is an online tool for selecting good color schemes for maps and other graphics.

Documentation

The manual, *An Introduction to R*, is part of the R installation; its graphics section is highly recommended. The Contributed Documentation section at the R website has several introductory manuals; *R for Beginners* by Emmanuel Paradis and *Using R for Data Analysis and Graphics* by John Maindonald have good chapters on graphics. Volume 2/2 of *R News* contains a short introduction to the lattice package.

Books

- *Introductory Statistics with R*
by Peter Dalgaard
Publisher: Springer Verlag
ISBN: 0387954759
Publication Date: August 2002
List Price: \$47.95
Paperback: 288 pages

— Excellent for getting started with R. Contains introductory material on graphics.

- *Modern Applied Statistics with S*, 4th edition
by Brian D. Ripley and William N. Venables
Publisher: Springer Verlag
ISBN: 0387954570
Publication Date: July 2002
List Price: \$72.95
Hardcover: 512 pages

— Intermediate-level text that includes many state-of-the-art methods. Chapter 4 discusses graphics.

- *The Elements of Graphing Data (Revised Edition)*
by William S. Cleveland
Publisher: Hobart Press
ISBN: 0963488414
Publication Date: September 1994
List Price: \$45.00
Hardcover: 297 pages

— Highly recommended. R uses many of Cleveland's principles by default.

Some graphical principles

The following list reproduces many of the points made in Cleveland's book.

- Make the data stand out. Avoid superfluity.
- Use visually prominent graphical elements to show the data.
- Make the data rectangle slightly smaller than the scale-line rectangle. Tick marks should point outward.
The data rectangle is the smallest rectangle that contains the data. The scale-line rectangle is the box that is drawn around the data.
- Use a pair of scale lines for each variable.
This is not the default in R, but can be easily done.
- Do not clutter the interior of the scale-line rectangle.
- Do not overdo the number of tick marks.
- Use a reference line when there is an important value that must be seen across the entire graph, but do not let the line interfere with the data.
- Do not allow data labels in the interior of the scale-line rectangle to interfere with the quantitative data or to clutter the graph.
- Avoid putting notes and keys inside the scale-line rectangle. Put a key outside, and put notes in the caption or in the text.
- Overlapping plotting symbols must be visually distinguishable.
- Superposed data sets must be readily visually assembled.
- Visual clarity must be preserved under reduction and reproduction.
- Put major conclusions into graphical form. Make captions comprehensive and informative.
 - Describe everything that is graphed.
 - Draw attention to the important features of the data.

- Describe the conclusions that are drawn from the data on the graph.
- Error bars should be clearly explained.
Sample standard deviation, standard error, confidence interval
- When logarithms of a variable are graphed, the scale label should correspond to the tick mark labels.
- Proofread graphs.
- Strive for clarity.
- When the orientations of line segments are judged to decode information about rate of change, bank the segments to 45°.
- Choose the range of the tick marks to include or nearly include the range of the data.
- Subject to the constraints that scales have, choose the scales so that the data rectangle fills up as much of the scale-line rectangle as possible.
- It is sometimes helpful to use the pair of scale lines for a variable to show two different scales.
- Do not insist that zero always be included on a scale showing magnitude.
- Use a logarithmic scale when it is important to understand percent change or multiplicative factors.
- Showing data on a logarithmic scale can cure skewness toward large values.
- Use a scale break only when necessary. If a break cannot be avoided, use a full scale break. Do not connect numerical values on two sides of a break. Taking logs may cure the need for a break.

Some other do's and don't's.

- Don't use 3-D bar plots.
- Don't use pie charts.
- Do use dot plots instead of bar plots.
- Do minimize ink-to-data ratio.
- Do enclose data rectangle with scale lines.
- Do make notations and symbols in a plot as consistent as possible with other plots, tables, and text.

Plot regions

Figure 1 illustrates the regions of a single R plot. If there is more than one plot per page, there is also an *outer margin*. The margins of a single plot contain tick marks, axis labels, and titles.

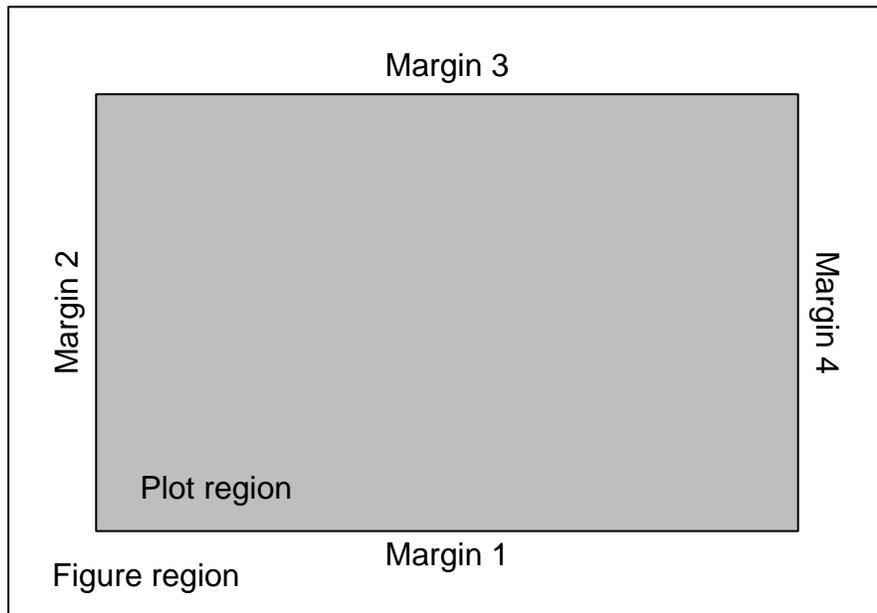
Figure 1 was produced using the following code:

```

postscript(file="R:/R Graphics Course/margins.ps", horizontal=F, width=5, height=3.5)
op ← par(no.readonly=TRUE)
par(mar=rep(0.1,4))
plot(1:10, type="n", xlab="", ylab="", axes=FALSE, xlim=c(0,10), ylim=c(0,7))
rect(0, 0, 10, 7)
rect(1, 1, 9, 6, col="gray")
text(0.5, 0.5, labels="Figure region", adj=c(0,0.5))
text(1.5, 1.5, labels="Plot region", adj=c(0,0.5))
text(5, 0.7, labels="Margin 1")

```

Figure 1: Regions of a single R plot.



```
text(0.7, 3.5, labels="Margin 2", srt=90)
text(5, 6.3, labels="Margin 3")
text(9.3, 3.5, labels="Margin 4", srt=270)
par(op)
dev.off()
```

Getting started

The following libraries will be used throughout:

```
library(ISwR)
library(Hmisc)
library(lattice)
```

```
old.par ← par(no.readonly=TRUE)
demo(graphics)
demo(image)
demo(persp)
par(old.par)
```

```
data(malaria)
attach(malaria)
plot(age, ab)
```

```
plotColor ← character(length(mal))
plotColor[mal==0] ← "red"
plotColor[mal==1] ← "blue"
plot(age, ab, col=plotColor)
```

```

jage ← jitter(age) # Or jitter2() from Hmisc
plot(jage, ab, type="n", xlab="", ylab="", axes=FALSE)
abline(h=500, col="gray80", lwd=2)
points(jage, ab, bg=plotColor, pch=21, cex=1.5)
title(xlab="Age (yrs)", ylab="Antibody level", cex.lab=1.2)
axis(1, lwd=2, font.axis=2)
axis(2, lwd=2, font.axis=2)
box(lwd=2)
axis(3, lwd=2, labels=FALSE)
axis(4, lwd=2, labels=FALSE)

detach(malaria)

coplot(ab ~ age | factor(mal), data=malaria)

```

Sunflower plots

```

data(iris)
sunflowerplot(iris[, 3:4])

```

identify()

```

data(state)
?state
ls()
income ← state.x77[, "Income"]
ill ← state.x77[, "Illiteracy"]
plot(income, ill)
identify(income, ill, labels=state.abb, n=3)

```

Polygons

```

curve(dnorm(x), -3, 3)

x ← seq(-3, 3, 0.02)
y ← dnorm(x)

which(pnorm(x) <= 0.05)
which(pnorm(x) >= 0.95)

plot(x, y, type="n")
polygon(c(x[1], x[1:68], x[68]), c(0, dnorm(x[1:68]), 0), col="blue", border="blue")
polygon(c(x[234], x[234:301], x[301]), c(0, dnorm(x[234:301]), 0), col="blue", border="blue")
points(x, y, type="l")
lines(c(0,0), c(0, dnorm(0)), lty=2)

arrows(-1.71, 0.105, -2.14, 0.13, length=0.2, code=1, lwd=2)

```

Single variable summaries

```

x ← rnorm(50)
boxplot(x, horizontal=T, notch=TRUE, boxwex=0.3, col="lightgray")

```

```
stripchart(x, vertical=F, add=T, at=1.2)
rug(x)
```

Histograms

```
xd ← density(x)
hist(x, prob=T)
points(xd$x, xd$y, type="l")

op ← par(no.readonly=TRUE)
par(fg="gray")
hist(x, prob=T, main="", col="yellow",
     xlab="", col.axis="gray", col.lab="gray", col.main="gray")
box()
axis(4, labels=FALSE)
par(op)

histbackback(rnorm(100), rnorm(10000), prob=T, brks=seq(-5,5,0.2))

histogram(~ ab | as.factor(mal), data=malaria)
```

Empirical cumulative distribution functions

```
groupA ← rnorm(n=30, mean=50, sd=10)
groupB ← rnorm(n=30, mean=40, sd=5)
ecdf(groupA, subtitles=FALSE)
ecdf(groupB, subtitles=FALSE, add=TRUE, lwd=2, col="gray80")
scat1d(groupA)
scat1d(groupB, side=1, col="gray80")

x ← locator(2)
legend(x, legend=c("GroupA", "GroupB"), col=c("black", "gray80"), lwd=1:2)
```

Dot plots

```
x ← sort(state.x77[sample(25),2])

lci ← x - runif(25, min=100, max=300)
uci ← x + runif(25, min=100, max=300)

dotchart(x, xlim=c(min(lci),max(uci)), pch=19)
axis(3, labels=FALSE)
title("Per Capita Income (1974)")
segments(lci, 1:25, uci, 1:25)
```

Color functions

```
colors()
?gray
?rgb
?hsv
?rainbow
```

The following graphic shows the named colors in R. To get the corresponding color names, use `colors()[n]`, where n is the number of the color in the chart. For example, `colors()[513]` yields “palegoldenrod”.

```
op ← par(no.readonly=TRUE)
par(bg="blue",fg="white",mar=c(1,2,1,2))
x ← c(rep(1:25,26),1:7)
y ← c(rep(27:2,rep(25,26)),rep(1,7))
plot(x,y,type="n",axes=FALSE,xlab="",ylab="",ylim=c(0,28), xlim=c(-0.5,27))
points(x,y,pch=15,cex=2,col=colors())
text(-0.1,27:1,labels=as.character(seq(1,651,25)),adj=1,cex=0.8)
text(26.2,27:2,labels=as.character(seq(25,650,25)),adj=0,cex=0.8)
lines(c( 5.5, 5.5),c(-0.3,28.3))
lines(c(10.5,10.5),c(-0.3,28.3))
lines(c(15.5,15.5),c(-0.3,28.3))
lines(c(20.5,20.5),c(-0.3,28.3))
par(op)
```

Star plots

```
data(mtcars)
palette(rainbow(12, s = 0.6, v = 0.75))
stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5),
      main = "Motor Trend Cars", draw.segments = TRUE)
```

Symbol plots

```
x ← 1:10
y ← sort(10*runif(10))
z ← runif(10)
symbols(x, y, thermometers=cbind(.5, 1, z), inches=.5, fg = 1:10)
```

Wireframes

```
bg ← trellis.par.get("background")
bg$col ← "white"
trellis.par.set("background", bg)

load("R:/R Graphics Course/eff1.df")
wireframe(eff ~ g1*g2, data=eff1.df,
          scales=list(cex = 0.8, distance = rep(1, 3), arrows = FALSE),
          xlab=expression(gamma[1]), ylab=expression(gamma[2]), zlab="",
          xlim=c(0,0.5), ylim=c(0,0.5), zlim=c(0,0.05), bg="white")
```

In addition to other functions in the lattice library (see `?Lattice`), the following functions are available for multivariate plots: `contour()`, `filled.contour()`, `image()`, and `persp()`.

Multiple graphs

```
op ← par(no.readonly=TRUE)
par(mar=rep(0,4), mfrow=c(2,2), oma=c(1,1,1,1)) # Or mfcol
plot(1:10, type="n", xlab="", ylab="", axes=FALSE, xlim=c(0,10), ylim=c(0,7)); box()
text(5, 3.5, labels="1", cex=1.2)
plot(1:10, type="n", xlab="", ylab="", axes=FALSE, xlim=c(0,10), ylim=c(0,7)); box()
text(5, 3.5, labels="2", cex=1.2)
```

```

plot(1:10, type="n", xlab="", ylab="", axes=FALSE, xlim=c(0,10), ylim=c(0,7)); box()
text(5, 3.5, labels="3", cex=1.2)
plot(1:10, type="n", xlab="", ylab="", axes=FALSE, xlim=c(0,10), ylim=c(0,7)); box()
text(5, 3.5, labels="4", cex=1.2)
mtext("Contiguous plots", outer=T, cex=2, line=3, side=3)
par(op)

x ← matrix(1:4,2,2)
layout(x)
layout.show(4)

x ← matrix(c(1:3,3),2,2)
layout(x)
layout.show(4)

layout(matrix(1:2,1,2), widths=c(3,1))
layout.show(2)

layout(matrix(1:2,2,1), heights=c(3,1))
layout.show(2)

# split.screen() can also be used.

```

Plot output

Use ?Devices to see a list of graphics devices currently in R.

Figure 2 was produced using the following code:

```

postscript(file="R:/R Graphics Course/chars.ps", horizontal=F, width=5, height=5)
op ← par(no.readonly=TRUE)
par(mar=rep(0.1,4))
x ← c(3,rep(1:5,5))
y ← c(6,rep(5:1,rep(5,5)))
z ← y + 0.4
plot(x, y, pch=0:25, type="n", axes=FALSE, xlab="", ylab="", ylim=c(1,6.7))
points(x, y, pch=0:25, cex=2, col="blue", bg="yellow")
text(x, z, labels=as.character(0:25), cex=1.2)
par(op)
dev.off()

```

Postscript files produce publication-quality graphics on laser printers, and they can be used in L^AT_EX documents. To create an encapsulated postscript file that can be imported into MS Word, use the following line:

```

postscript(file="R:/R Graphics Course/Fig2.eps", horizontal=FALSE, onefile=FALSE,
  paper="special", width=5, height=6)

```

It is important not to resize the graphic within Word. If you need a different size, use different width and height arguments in the postscript() function.

Although other forms of graphic can be imported into Word, they do not usually print well.

The following code produces a graphic that can be imported into MS Powerpoint:

```
win.metafile("R:/R Graphics Course/boxplot.wmf")
op ← par(no.readonly=TRUE)
line.col ← "gray"
data(kfm)
boxplot(weight ~ sex, data=kfm, boxwex=0.3, ylab="", names=c("Boys","Girls"),
        notch=TRUE, col=heat.colors(2), border=line.col,
        pars=par(fg=line.col, col.axis=line.col, col.lab=line.col, cex=1.5, lwd=2))
mtext("Weight (kg)", side=2, line=2.5, cex=1.5)
par(op)
dev.off()
```

Figure 2: R plotting symbols.

